Анализ кода на наличие запахов

1. Дублирование кода

- **Описание**: Дублирование кода возникает, когда один и тот же фрагмент кода повторяется в нескольких местах.
- **Пример**: В данном коде дублирования нет, так как алгоритм Флойда-Уоршелла и основная логика программы выделены в отдельные методы.
- **Решение**: Если в будущем появится необходимость использовать один и тот же код в нескольких местах, его следует вынести в отдельный метод.

2. Длинные методы

- **Описание**: Методы, выполняющие слишком много операций, могут быть сложными для понимания и тестирования.
- **Пример**: Метод Main выполняет несколько задач: инициализацию данных, взаимодействие с пользователем и вычисление расхода топлива.
- **Решение**: Разделить метод Main на более мелкие методы, каждый из которых будет выполнять одну задачу. Например, можно создать методы InitializeGraph, GetUserInput, CalculateFuelConsumption.

3. Магические числа

- **Описание**: Магические числа — это константы, значение которых не очевидно из контекста.
- **Пример**: В коде используются числа 0.94, 1.88 и т.д. для обозначения длины путей.
- **Решение**: Вынести эти значения в константы с осмысленными именами, например, ROAD_LENGTH_1_2 = 0.94.

4. Жесткая связанность (Hardcoding)

- **Описание**: Жесткая связанность возникает, когда значения или логика задаются непосредственно в коде.
- **Пример**: Матрица смежности задана непосредственно в коде.
- **Решение**: Если в будущем потребуется изменять данные о путях, их можно вынести в отдельный конфигурационный файл или базу данных.

Заключение

Анализ кода показал, что в программе присутствуют некоторые запахи кода, которые могут затруднить её поддержку и расширение. Рекомендуется разделить длинные методы, вынести

магические числа в константы, добавить обработку ошибок и рассмотреть возможность вынесения данных о путях в отдельный конфигурационный файл. Эти изменения помогут улучшить читаемость и поддерживаемость кода.